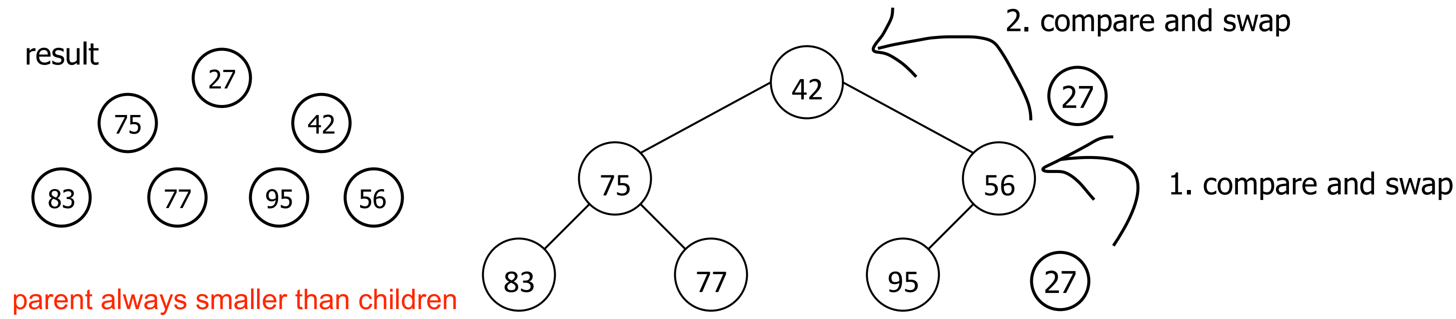
CS526 O2
Homework Assignment 4

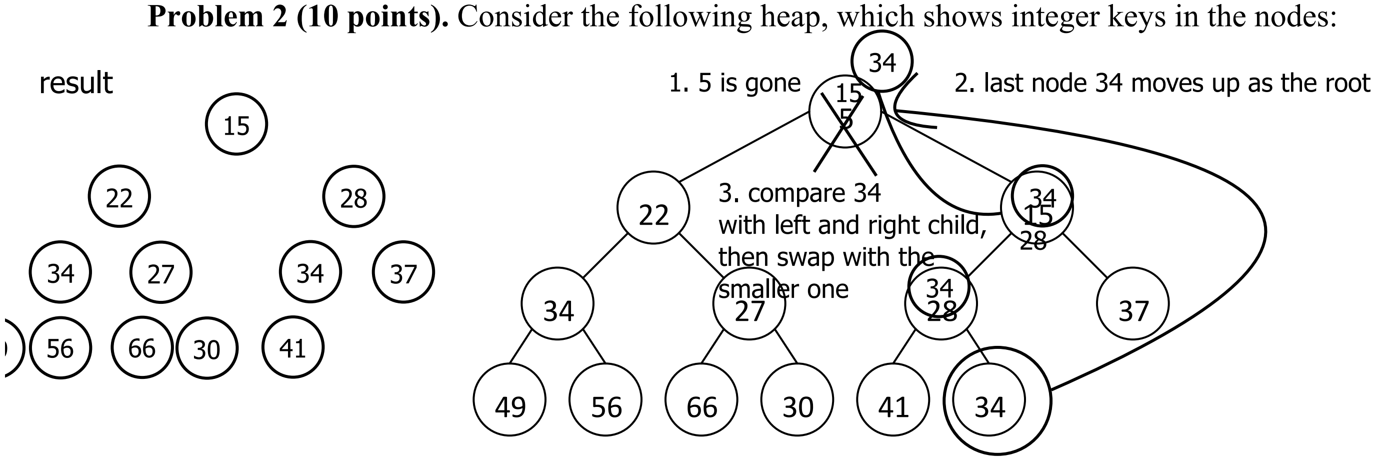
Problem 1 (10 points). Consider the following heap, which shows integer keys in the nodes:



Show the resulting tree if you add an entry with key = 27 to the above tree? You need to describe, step by step, how the resulting tree is generated.

First, 27 will be added to the end, then we reorganize the heap to maintain heap-order property which the parent key should always be smaller or equal to the child key. When adding an element to the heap, it uses up-heap bubbling that bubbling all the way back to the root by finding last empty location, swap based on the priority with the parent, until we get the point that the child is no longer smaller than parent. Therefore, 27 compares with its parent 56, since 27 is smaller, so it swaps with 56, then it compares with the root 42, it is smaller than 42, so it swaps with 42. Now, 27 will become the new root of the heap.

Problem 2 (10 points). Consider the following heap, which shows integer keys in the nodes:



remove min and last node as root then compare

Suppose that you execute the *removeMin*( ) operation on the above tree. Show the resulting tree. You need to describe, step by step, how the resulting tree is generated.

<5, 22, 15, 34, 27, 28, 37, 49, 56, 66, 30, 41, 34> - initial, then we execute removeMin() where we remove the root, 5 is removed, last entry 34 moves up to be the new root and performs down-heap bubbling, where each time we compare the current node with both left and right child, and replace it with the minimum child. As 34 moves up, it compares 22 and 15, 15 is smaller so 15 moves up, then 34 compares with its current children 28 and 37, where 28 is smaller so 28 moves up, then 34 is smaller then 41 so it can stays with where it is now. The result should be <15, 22, 28, 34, 27, 34, 37, 49, 56, 66, 30, 41> now.

Problem 3 (10 points). This problem is about the chaining method we discussed in the class. Consider a hash table of size N = 11. Suppose that you insert the following sequence of keys to an initially empty hash table. Show, step by step, the content of the hash table.

Sequence of keys to be inserted: <5, 8, 44, 23, 12, 20, 35, 32, 14, 16>

0	1	2	3	4	5	6	7	8	9	10
44	23	35	14		5			8	20	32
	12				16					

chaining, mod, a mod b, if a < b, hash key = a

5 mod 11 = 5 - first in address 5
8 mod 11 = 8
44 mod 11 = 0
23 mod 11 = 1 - first in address 1
12 mod 11 = 1 - placed below 23
20 mod 11 = 9
35 mod 11 = 2
32 mod 11 = 10
14 mod 11 = 3
16 mod 11 = 5 - placed below 5

Problem 4 (10 points). This problem is about linear probing method we discussed in the class. Consider a hash table of size N = 11. Suppose that you insert the following sequence of keys to an initially empty hash table. Show, step by step, the content of the hash table.

Sequence of keys to be inserted: <5, 8, 44, 23, 12, 20, 35, 32, 14, 16>

0	1	2	3	4	5	6	7	8	9	10
44	23	12	35	14	5	16		8	20	32

probing => move next open one

5 mod 11 = 5
8 mod 11 = 8
44 mod 11 = 0
23 mod 11 = 1
12 mod 11 = 1 - move to index 2
20 mod 11 = 9
35 mod 11 = 2 - move to index 3
32 mod 11 = 10
14 mod 11 = 3 - move to index 4
16 mod 11 = 5 - move to index 6

Problem 5 (10 points). Suppose that your hash function resolves collisions using open addressing with double hashing, which we discussed in the class. The double hashing method uses two hash functions h and h'.

Assume that the table size $N = 13$, $h(k) = k \bmod 13$, $h'(k) = 1 + (k \bmod 11)$, and the current content of the hash table is:

open address: h(k) = k mod n (total number), h'(k) = 1 + (k mod num)

0	1	2	3	4	5	6	7	8	9	10	11	12
	16	2	29		18			21	48	15		

If you insert $k = 16$ to this hash table, where will it be placed in the hash table? You must describe, step by step, how the location of the key is determined.

h(k) = k mod 13
h(16) = 16 mod 13 = 3

h'(k) = 1 + (k mod 11)
h'(16) = 1 + (16 mod 11) = 6

h(k, i) = (h(k) + i*h'(k)) mod N
i = 0;
[3+0(6)]mod13=3 => index occupied
i = 1;
[3+1(6)]mod13=9 => index occupied
i = 2;
[3+2(6)]mod13=2 => index occupied
i = 3;
[3+3(6)]mod13=8 => index occupied
i = 4;
[3+4(6)]mod13=1 => probe

h(k, i) = (h(k) + i*h'(k)) mod N